Programming with FORTRAN

I. Objective

To learn the basic commands needed to write FORTRAN programs.

II. Introduction

Up to this point we have seen that many interesting physics problems may be solved within available software packages without any programming at all. However, there are several occasions when obtaining a numerical solution requires one to write a program:

- When the problem is very specialized. Research often presents problems that are very specific in nature for which software packages do not exist.
- When the problem is too big for existing software. The solution of a 100 x 100 set of coupled linear equations would typically be too big for Maple. In this case writing a program would make sense.
- When the problem requires high computation speed. A program dedicated to a single task will typically run faster than a general application designed to handle many different problems.

In today's exercises we will extend what you learned in Physics 221 for wave motion in a wire of uniform mass density to the case of a non-uniform wire We will solve the problem numerically by writing a program in FORTRAN on the Project Vincent workstations, and calculate the frequencies and wave functions of the resonant standing wave modes.. We begin by showing some essential operations in the UNIX environment.

III. Exercises

A. Running Programs in the UNIX Environment (Optional)

Objective:	to gain familiarity with compiling and running FORTRAN programs, and graphing data files on Project Vincent
Where to begin:	on Project Vincent
What to do:	follow the instructions below
What to turn in to your instructor:	nothing: for your information only
What to put in log book:	any problems you have and how you solve them

(1) Copy Files from the Physics 232 Locker: We placed several files you may use for today's activities in the Physics 232 locker. Go to the locker (to access the locker type **add physics** and then go to the directory **/home/physics/phys232**). Copy the files **in, aaa, numerov.f,** and **test.f** to your working directory.

(2) Running FORTRAN Programs: Suppose we wish to run the FORTRAN program **test.f** (note that FORTRAN program names must end with the extension **.f**) that reads a variable from the input file **aaa**. First the program must be **compiled**. This is done by typing: **f77 -o test test.f**

Command	What it does
f77 -o test test.f	compiles the FORTRAN program **test.f** and creates and executable file called **test**

The program **test** calculates the quantity aN^2, where *a* is a number specified in the input file **aaa** and *N* is an integer from 0 to 10. Set *a* to the value 1 by editing **aaa** using the command: **joe aaa:**

Command	What it does
joe aaa	uses the **joe** editor to open the file **aaa**. The editor commands may be displayed by typing **^k^h (control-k control-h)**

To run the program type: **test<aaa**:

Command	What it does
test<aaa	runs the executable file **test** which reads the input file **aaa**

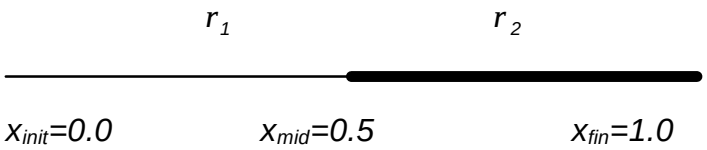
(3) Graphing Data Files: The output of the program was written to a file called **fort.8.** To graph the contents of this two column file type: **xgraph fort.8**:

Command	What it does
xgraph fort.8	graphs the data in **fort.8**

B. Waves in a Wire

Objective:	to outline the physics of today's problem
What to do:	read the material below and derive the stated equations
What to turn in to your instructor:	your log book
What to put in log book:	derivations from this exercise

(1) Vibrations of a Non-uniform Wire: Suppose that two pieces of wire of equal length with masses per unit length r_1 and r_2 are joined to form one piece of total length *L=1 meter.* The wire is then placed under a tension *T* and its ends are fixed, as shown in the figure. Suppose that r_1*=0.001 kg/m*, r_2 *0.002 kg/m*, *L=1 m,* and *T=100 N* . Suppose also that our coordinates are as shown



in the Figure above. We wish to study the resonant standing waves in this wire and determine the frequencies of the vibrational modes.

(2) Finding Vibrational Frequencies Using the Shooting Method: In Physics 221 you learned that a wave traveling in a uniform wire of linear mass density r under a constant tension T has a frequency given by

$$f = \frac{k}{2p}\sqrt{\frac{T}{r}}$$

where the wave number k is defined as

$$k = \frac{2p}{l}$$

The transverse displacement y of the wire as a function of position x is given by

$$\frac{d^2y}{dx^2} + k^2 y = 0$$

If the wire is fixed at both ends then one has the **boundary conditions** $y(0)=y(L)=0$ and one finds that resonant standing waves in the wire occur for discrete values of k, and therefore discrete frequencies $f = \frac{k}{2p}\sqrt{\frac{T}{r}}$. If one wants to numerically determine the values of k for the resonant standing waves occur, one could start with the two boundary conditions at the left end of the wire, $y(0)$ and $y'(0)$, *guess an approximate value of k,* numerically solve for the function y over the interval [0,L], and then check to see how close $y(L)$ was to 0, the desire boundary condition. If the difference were not acceptable, a new guess for k could be made and a new $y(L)$ calculated until $y(L) \approx 0$ within an acceptable tolerance. This trial and error approach is called the **shooting method** and is the method we will use to determine the resonant frequencies of a non-uniform wire.

(3) Defining the Problem Mathematically: We therefore wish to solve the differential equation

$$\frac{d^2y}{dx^2} + k^2 y = 0$$

for the case of the non-uniform wire, where k is a function that may be written as

$$k(x) = \begin{cases} k_1 \text{ if } 0 \le x \le \frac{1}{2} \\ k_1\sqrt{\frac{r_2}{r_1}} \text{ if } \frac{1}{2} < x \le 1 \end{cases}$$

We will use the boundary values $y(0)$ and $y'(0)$, along with an initial guess for k_1, to solve for the values of y over the interval [0,1].

(4) Derivation: Show in your log book that, because the two parts of the wire vibrate with the same frequency, the following formula holds:

$$\frac{k_2}{k_1} = \sqrt{\frac{r_2}{r_1}}$$

C. Numerical Solution Using Numerov Algorithm

Objective:	to set up the algorithm we will use to solve the differential equation $\frac{d^2y}{dx^2}+k^2y=0$
Where to begin:	here
What to do:	follow the instructions below
What to turn in to your instructor:	your log book
What to put in log book:	derivation of the equations specified below

(1) The Algorithm: Once the differential equation and the boundary condition at some point is determined, the differential equation may be solved numerically. Differential equations of the general form

$$\frac{d^2y}{dx^2}+k^2(x)y=S(x)$$

may be solved using the **Numerov algorithm** (see Appendix 1 for the recursion formulas). In your log book, show that the Numerov recursion formulas for the case of *S(x)=0* become

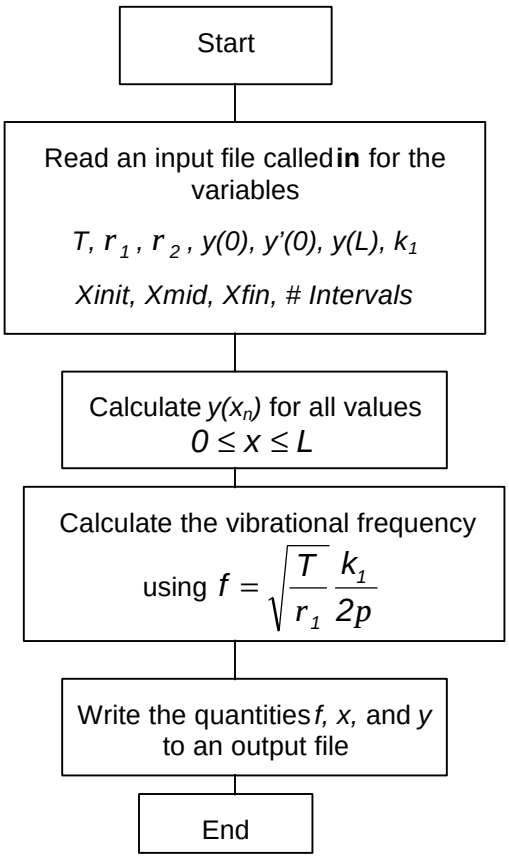
$$y_1=y_0\left(1-\frac{h^2k_0^2}{2}\right)+y_0'h$$

$$y_i=\frac{1}{\left(1+\frac{h^2k_i^2}{12}\right)}\left\{2\left(1-\frac{5h^2k_{i-1}^2}{12}\right)y_{i-1}-\left(1+\frac{h^2k_{i-2}^2}{12}\right)y_{i-2}\right\}$$

D. Writing the Program

Objective:	to write a FORTRAN program to numerically determine the frequencies and wave functions for the resonant standing waves in a non-uniform wire
Where to begin:	on Vincent
What to do:	follow the instructions below
What to turn in to your instructor:	your log book
What to put in log book:	the time you begin your work, problems you encounter, solutions you find

(1) Planning the Program: The program you write should have the following general outline:



(2) Creating the Main Program: To begin writing the main program, type the command:

joe num.f

Command	What it does
joe	begins an editing program called **joe**
num.f	creates a file called **num.f**. Note that FORTRAN programs must have the extension **.f**

Now we begin to write the program in FORTRAN. Keep in mind the following general rules for programming in FORTRAN:

FORTRAN Rules
A letter **c** in the first space of a line indicates that a comment will follow. FORTRAN will skip the line
Array dimensions must be specified at the beginning of a FORTRAN.
Variable types must be declared at the beginning of a FORTRAN program. Undeclared variable names that begin with letters a-h and o-z are understood to be real and single precision. Variable names beginning with letters i-n are understood to be integers.
Indent all FORTRAN commands at least eight spaces (except for when numbering lines)

Type the following lines:

```
ccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c      Numerov Algorithm
c
ccccccccccccccccccccccccccccccccccccccccccccccccccc
      dimension x(10000),y(10000)
      character a
      common/one/ xk1,xk2,xinit,xmid,xfin
```

Command	What it does
dimension x(10000),y(10000)	creates two arrays **x** and **y** of dimension 1x10000 (i.e. a vector with 10000 components)
character a	creates a variable **a** that contains characters (e.g. letters and symbols)
common /one/	makes the variables following the **common** command available to subroutines in the program that include the same **common** statement. The symbol **/one/** is a label (any other name will do as long as it begins with a letter) that distinguishes different **common** statements (though in this program there is only one).

(3) Reading the Input File: The input file **in** contains all of the information about the physical parameters, boundary conditions, and the number line. If you view the contents of the file (type **more in**), you will see several lines of text separating lines of numbers. The text in the file serves only to remind us what the numbers represent. In reading the file, therefore, we wish only to extract the numbers. To do this type the following lines:

```
      read(5,10) a,a,a
      read(5,*) T,rho1,rho2
      read(5,10) a,a,a,a
      read(5,*) y0,Dy0,yL
      read(5,10) a,a,a,a
      read(5,*) xk1
      read(5,10) a,a,a,a
      read(5,*) xinit,xmid,xfin,N
10    format(a1)
```

Command	What it does
read(5,10) a,a,a **10 format(a1)**	opens the input file (which is designated by the **5**), and reads three character strings that are one character long (specified by the symbol **a1**). This is how the text lines are skipped.
read(5,*) T,rho1,rho2	reads the numbers and assigns them to the variables T, rho1, and rho2

(4) Performing the Calculations: We now wish to write the part of the program that calculates the function *y(x),* the frequency *f*, and writes the values of *x* and *y* to an output file. Type the following:

```
xk2=xk1*sqrt(rho2/rho1)
pi=3.1415926
ymax=0.
call numerov(xinit,y0,Dy0,xfin,N,ymax,x,y)

delta=(y(N)-yL)/ymax
freq=sqrt(T/rho1)*xk1/2/pi

write(7,*) 'Delta = ',delta
write(7,*) 'Frequency = ',freq
do i=0,N
      write(7,*) x(i),y(i)
enddo
end
```

Command	What it does
call numerov( )	calls a subroutine **numerov** that performs the calculations of the Numerov algorithm. The variables **xinit, y0,Dy0,xfin,** and **N** are known at the time of the call, and are passed to the subroutine. The variables **ymax, x, y** are calculated by the subroutine and passed back to the main program. Note **x** and **y** are arrays.
delta=(y(N)-yL)/ymax	calculates by how much the final value of **y** misses the specified boundary condition **yL** as a fraction of the maximum value of the wave function
freq=sqrt(T/rho1)*xk1/2/pi	calculates the vibration frequency from the relation $f = \sqrt{\frac{T}{r_1}}\frac{k_1}{2p}$
write(7,*) 'Delta=',delta	writes the value of **delta** to an output file that FORTRAN will call **fort.7**
do i=0,N **write(7,*)** **x(i),y(i)** **enddo**	a **do loop** that writes the values of **x** and **y** to the file **fort.7** for values of the index **i** from 0 to N (the indentation in front of the **write** is not necessary, but makes the program easier to read).
end	signals that the main program has ended; defines the boundary of the program to which the variable names apply

(5) Writing the Numerov Subroutine: We now wish to write the subroutine that performs the recursion relation of the Numerov algorithm. The subroutine is written to accept the initial vales **xi, yi, Dyi, xf, N** and to calculate **x** and **y** and determine the maximum y value **ymax**. One way to write the subroutine is as follows:

```
ccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine numerov(xi,yi,Dyi,xf,N,ymax,x,y)
      dimension x(10000),y(10000)

      dx=(xf-xi)/N

      do i=0,N
            if(i.eq.0) then
                  x(0)=xi
                  y(0)=yi
                  if (abs(y(0)).gt.ymax) ymax=abs(y(0))
            else if(i.eq.1) then
                  x(1)=x(0)+dx
                  xk=wav(x(0))
                  y(1)=yi*(1-dx*dx*xk*xk/2)+Dyi*dx
                  if (abs(y(1)).gt.ymax) ymax=abs(y(1))
             else
                  x(i)=x(i-1)+dx
                  xk0=wav(x(i))
                  xk1=wav(x(i)-dx)
                  xk2=wav(x(i)-2*dx)
                  y(i)=1/(1+dx*dx*xk0*xk0/12)*
     +                  (2*(1-5*dx*dx*xk1*xk1/12)*y(i-1)
     +                  -(1+dx*dx*xk2*xk2/12)*y(i-2))
                  if (abs(y(i)).gt.ymax) ymax=abs(y(i))
            endif
      enddo
      end
```

Command	What it does
subroutine numerov ( )	begins the subroutine **numerov**
dimension x(10000),y(10000)	dimension statements are needed in all subroutines that use arrays

if(i.eq.0) then
(process A)
else if(i.eq.1) then
(process B)
else
(process C)
endif
	this is a **conditional**. If i=0 then process A is performed and the program goes to **endif.** If i=1 then process B is performed and the program goes to **endif**. Otherwise process C is performed and the program goes to **endif**

xk=wav(x(i))	determines the value of the wave number **wav** at the point **x(i)** and assigns the number to the variable **xk**
y(i)=1/(1+dx*dx*xk0*xk0/12)* **+ (2*(1-5*dx*dx*xk1*xk1/12)*y(i-1)**	calculates the Numerov recursion relation. Note that when a formula will not fit on one line, it may be wrapped over several lines by placing any desired

+ -(1+dx*dx*xk2*xk2/12)*y(i-2))	character (in this case **+**) in the sixth column

Command	What it does
dx*dx	note that I have used **dx*dx** instead of **dx^2**. Though this is splitting hairs, for UNIX compilers the first is calculated more quickly than the second.
return **end**	all subroutines in FORTRAN must end with these lines

(6) Writing the Function Subroutine: The last task is to define the function *k(x).* This will be calculated in the function subroutine **wav.** One way to do this is shown below:

```
ccccccccccccccccccccccccccccccccccccccccccccccccccc
      function wav(x)
      common/one/ a,b,c,d,e

      if (x.ge.c.and.x.le.d) then
            wav=a
      else if (x.ge.d.and.x.le.e) then
            wav=b
      endif
      return
      end
```

Command	What it does
function wav (x)	begins the function subroutine **wav**
common/one/ a,b,c,d,e	makes the variables **xk1, xk2, xinit, xmid, xfin** in the main routine common to the function subroutine. The assignment is in the order listed so that **a=xk1, b=xk2, c=xinit, d=xmid, e=xfin**
return **end**	all subroutines in FORTRAN must end with these lines

E. Running the Program

Objective:	to determine the frequencies and wave functions for the resonant standing waves in a non-uniform wire
Where to begin:	on Vincent
What to do:	follow the instructions below
What to turn in to your instructor:	your log book, including the standing wave frequencies you determine, graphs of the modes of vibration
What to put in log book:	the time you begin your work, problems you encounter, solutions you find

(1) Checking the Results: There are several ways the program may be checked. First, if the wire were uniform, you learned in Physics 221 that the standing wave will have *k*

values given by $k_n = np$ and $y(x)=\sin(k_n x)$. Make the necessary changes in the r and k values of the input file and check if you get the expected results.

Think of another check you can do to test your program. Comment on your results.

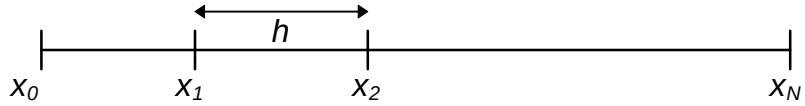
(2) Finding the Resonant Frequencies for the Non-uniform Wire: For $N=100$, run your program for the parameter values given in problem III.B.1. Use the shooting method described in III.B.2 to determine the first three resonant frequencies of the system. You may use **xgraph** to plot your results and see how close your guess for k_1 brought you to the final boundary condition. Print out the graphs for the first three standing wave modes.

IV. Appendix 1: The Numerov Algorithm

Equations of the form

$$\frac{d^2y}{dx^2} + k^2(x)y = S(x)$$

may be solved for $y(x)$ for points on the axis using the **Numerov Algorithm**[1] if the boundary conditions $y(x_0)$, $y'(x_0)$ are known at some point x_0.. The axis is divided into N intervals of equal length $h = \frac{x_N - x_0}{N}$ as shown below. This method may be summarized by the recursion



relations in the table below. We use the notation $y_i \equiv y(x_i)$, $y_i' = y'(x_i)$ and similar notation for k_i and S_i.

Numerov Algorithm Recursion Relations for y_0, y_1, and y_i
$$y_0 = y(x_0)$$
$$y_1 = y_0 + y_0' h + \frac{(-k_0^2 y_0 + S_0)}{2} h^2$$
$$y_i = \frac{1}{\left(1 + \frac{h^2 k_i^2}{12}\right)} \left\{ \frac{h^2}{12}(S_i + 10S_{i-1} + S_{i-2}) + 2\left(1 - \frac{5h^2 k_{i-1}^2}{12}\right) y_{i-1} - \left(1 + \frac{h^2 k_{i-2}^2}{12}\right) y_{i-2} \right\}$$

V. Appendix 2: FORTRAN Code for Numerov Program

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Numerov Algorithm
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          dimension x(10000),y(10000)
```

[1] See S. E. Koonin, *Computational Physics,* Chapter 3, for a complete description.

```
      character a
      common/one/ xk1,xk2,xinit,xmid,xfin

      read(5,10) a,a,a
      read(5,*) T,rho1,rho2
      read(5,10) a,a,a,a
      read(5,*) y0,Dy0,yL
      read(5,10) a,a,a,a
      read(5,*) xk1
      read(5,10) a,a,a,a
      read(5,*) xinit,xmid,xfin,N
10    format(a1)

      xk2=xk1*sqrt(rho2/rho1)
      pi=3.1415926
      ymax=0.
      call numerov(xinit,y0,Dy0,xfin,N,ymax,x,y)

      delta=(y(N)-yL)/ymax
      freq=sqrt(T/rho1)*xk1/2/pi
      write(7,*) 'Delta = ',delta
      write(7,*) 'Frequency = ',freq
      do i=0,N
            write(7,*) x(i),y(i)
      enddo
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine numerov(xi,yi,Dyi,xf,N,ymax,x,y)
      dimension x(10000),y(10000)

      dx=(xf-xi)/N
      do i=0,N
            if(i.eq.0) then
                  x(0)=xi
                  y(0)=yi
                  if (abs(y(0)).gt.ymax) ymax=abs(y(0))
            else if(i.eq.1) then
                  x(1)=x(0)+dx
                  xk=wav(x(0))
                  y(1)=yi*(1-dx*dx*xk*xk/2)+Dyi*dx
                  if (abs(y(1)).gt.ymax) ymax=abs(y(1))
             else
                  x(i)=x(i-1)+dx
                  xk0=wav(x(i))
                  xk1=wav(x(i)-dx)
                  xk2=wav(x(i)-2*dx)
                  y(i)=1/(1+dx*dx*xk0*xk0/12)*
     +                  (2*(1-5*dx*dx*xk1*xk1/12)*y(i-1)
     +                  -(1+dx*dx*xk2*xk2/12)*y(i-2))
                  if (abs(y(i)).gt.ymax) ymax=abs(y(i))
            endif
      enddo
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      function wav(x)
      common/one/ a,b,c,d,e
      if (x.ge.c.and.x.le.d) then
            wav=a
```

```
else if (x.ge.d.and.x.le.e) then
      wav=b
endif
return
end
```